
Introducción a Ruby on Rails

— Strings —

Overview

- Ruby soporta distintos tipos de string
- La API de Strings de ruby es super potente.
- Símbolos

Tipos de String

- Single-quote String (literal)
 - Permiten escapar el caracter ' con \
 - Muestran (casi todo) el resto de los caracteres tal como existen en el string.
 - Show (almost) everything else as is
- Double-quote String
 - Interpretan caracteres especiales como `\n` y `\t`
 - Permiten la interpolación de strings

Strings & Interpolación

```
literal = 'Oga ari, oga ari, oi ka\'i \n Chake Karaja!'  
string = "Oga ari, oga ari, oi ka\'i \n Chake Karaja!"
```

```
puts literal # => Oga ari, oga ari, oi ka'i \n Chake Karaja!  
puts string  # => Oga ari, oga ari, oi ka'i  
              # => Chake Karaja!
```

```
def multiplicar (uno, dos)  
  "#{uno} multiplicado por #{dos} es igual a #{uno * dos}"  
end
```

```
puts multiplicar(5, 3)  
# => 5 multiplicado por 3 es igual a 15
```



Interpolación de strings

Más strings

- Si un método termina con **!**, significa que modifica el string
 - Si no termina con **!** significa que retorna una copia del string con la modificación.
- Se puede utilizar **%Q** {para escribir strings de múltiples líneas}
 - Tiene el mismo comportamiento de un string double-quote (interpreta los caracteres especiales).

```
mi_nombre = " juan"
puts mi_nombre.lstrip.capitalize # => Juan
p mi_nombre # => " juan"
mi_nombre.lstrip! # (destructivo) Elimina el espacio
mi_nombre[0] = 'K' # Reemplaza el primer caracter
puts mi_nombre # => Kuan
```

```
clase_actual = %Q{Hoy no juega Paraguay...
                 Vamos a dar la clase completa!}
```

```
clase_actual.lines do |line|
  line.sub! 'Paraguay', 'Argentina'
  # reemplaza 'Paraguay' por 'Argentina'
  puts "#{line.strip}"
end
# => Hoy no juega Argentina...
# => Vamos a dar la clase completa!
```

Strings API

📄 ruby-doc.org/core-2.3.0/String.html

Home






Core

Std-lib

Downloads

Home Classes Methods

In Files

 complex.c
 pack.c
 rational.c
 string.c
 transcode.c

Parent

Object

Methods

String

A `String` object holds and manipulates an arbitrary sequence of bytes, typically representing characters. `String` objects may be created using `String::new` or as literals.

Because of aliasing issues, users of strings should be aware of the methods that modify the contents of a `String` object. Typically, methods with names ending in “!” modify their receiver, while those without a “!” return a new `String`. However, there are exceptions, such as `String#[]=`.

Public Class Methods

Símbolos

- :simbolo - Strings altamente optimizados
- Nombres de constantes que no se necesitan pre-declarar
- Un tipo de string que representa algo.
- Son únicas e inmutables
- Pueden ser convertidas a string para su manipulación mediante `to_s`
- O se puede convertir un string a symbol, con `to_sym`

Métodos

```
> :hello.methods
=> [:<=>, :==, :===, :=~, :[], :empty?, :inspect, :intern,
:length, :size, :succ, :to_sym, :to_proc, :to_s,
:casecmp, :match, :upcase, :downcase, :capitalize,
:slice, :encoding, :id2name, :<, :>, :<=, :>=, :
:to_json, :instance_of?, :public_send, :instance
:instance_variable_set, :instance_variable_defin
:remove_instance_variable, :private_methods, :ki
:instance_variables, :tap, :define_singleton_met
:public_method, :extend, :singleton_method, :to_
:!, :eql?, :respond_to?, :freeze, :display, :ob
:method, :nil?, :hash, :class, :singleton_class,
:itself, :taint, :tainted?, :untaint, :untrust,
:untrusted?, :methods, :protected_methods, :froz
:public_methods, :singleton_methods, :!, :!=, :-
:equal?, :instance_eval, :instance_exec, :__id__]
> "hello".methods
=> [:include?, :%, :*, :+, :unicode_normalize, :to_c,
:unicode_normalize!, :unicode_normalized?, :count, :partition,
:unpack, :encode, :encode!, :next, :casecmp, :insert, :bytesize,
:match, :succ!, :next!, :upto, :index, :rindex, :replace,
:clear, :chr, :+@, :-@, :setbyte, :getbyte, :<=>, :<<, :scrub,
:scrub!, :byteslice, :==, :===, :dump, :=~, :downcase, :[], :
[]=, :upcase, :downcase!, :capitalize, :swapcase, :upcase!,
:oct, :empty?, :eql?, :hex, :chars, :split, :capitalize!,
:swapcase!, :concat, :codepoints, :reverse, :lines, :bytes,
:prepend, :scan, :ord, :reverse!, :center, :sub, :freeze,
:inspect, :intern, :end_with?, :gsub, :chop, :crypt, :gsub!,
:start_with?, :rstrip, :sub!, :ljust, :length, :size, :strip!,
:succ, :rstrip!, :chomp, :strip, :rjust, :lstrip!, :tr!,
:chomp!, :squeeze, :lstrip, :tr_s!, :to_str, :to_sym, :chop!,
:each_byte, :each_char, :each_codepoint, :to_s, :to_i, :tr_s,
:delete, :encoding, :force_encoding, :sum, :delete!, :squeeze!,
:tr, :to_f, :valid_encoding?, :slice, :slice!, :rpartition,
```

Entonces..

- La interpolación de strings es algo bastante útil.
- La API de ruby para strings es muy potente.